PTO/SB/21 (09-04)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.



# TRANSMITTAL FORM

*(to be used for all correspondence after initial filing)*

Total Number of Pages in This Submission: 29 30

| | |
|---|---|
| Application Number | 09/591,225 |
| Filing Date | June 9, 2000 |
| First Named Inventor | David R. Baldwin |
| Art Unit | 2676 |
| Examiner Name | Tung, Kee M. |
| Attorney Docket Number | TD-155 |

## ENCLOSURES *(Check all that apply)*

- ☐ Fee Transmittal Form
  - ☐ Fee Attached
- ☐ Amendment/Reply
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Reply to Missing Parts/ Incomplete Application
  - ☐ Reply to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) _____
  - ☐ Landscape Table on CD
- ☐ After Allowance Communication to TC
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☑ Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- ☐ Proprietary Information
- ☐ Status Letter
- ☐ Other Enclosure(s) (please Identify below):

Check No. 1393 for $500.00, and Return Postcard

Remarks

Submission of Appeal Brief with Claims Appendix

## SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

| | | | |
|---|---|---|---|
| Firm Name | Groover & Holmes<br>Customer No. 29106 | | |
| Signature | N.E.Pham | | |
| Printed name | N. Elizabeth Pham | | |
| Date | April 22, 2005 | Reg. No. | 49,042 |

## CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

| | | | |
|---|---|---|---|
| Signature | Peggy Heath | | |
| Typed or printed name | Peggy Heath | Date | April 22, 2005 |

**In the United States Patent and Trademark Office**

In re application of: :

Baldwin : Art Unit: 2676

AN 09/591,225 : Examiner: Tung, Kee M.

Filed: 06/09/2000 : Attorney Docket No.: TD-155

For: Graphics Memory Management With Invisible Hardware-Managed Page Faulting (confirmation no. 3467)

APPEAL BRIEF

Honorable Commissioner of Patents and Trademarks

Alexandria, VA 22313

Sir:

Appellant respectfully submits that Examiner Tung's decision of 11/30/2004, finally rejecting claims 1-5, 7-13, 16, 18, and 19 in the present application, should be reversed in view of the following arguments and authorities.

TABLE OF CONTENTS

REAL PARTY IN INTEREST

The real party in interest, and assignee of the present application, is 3Dlabs Inc., Ltd., of Reid Hall, Hamilton HM11, Bermuda.

RELATED APPEALS AND INTERFERENCES

To the best knowledge and belief of the undersigned attorney, there are no related appeals or interferences.

STATUS OF CLAIMS

Claims 1-5 and 7-20 are pending in the present application. Claims 14, 15, 17, and 20 are allowed. Claim 6 has been canceled by a previous amendment. Claims 1-5, 7-13, 16, 18, and 19 have been rejected. Claims 1-5, 7-13, 16, 18, and 19 are the subject of this appeal.

STATUS OF AMENDMENTS

There have been no amendments after the final rejection mailed 11/30/2004.

SUMMARY OF CLAIMED SUBJECT MATTER

Five independent claims (1-4 and 7) are the subject of this appeal.

**Claim 1:**

**Language:**

A computer system, comprising: a graphics accelerator unit which manages page faulting of texture data invisibly to the host processor.

**Summary:**

In particular, the present application discloses a computer system in which a graphics accelerator unit manages page faulting of texture data invisibly to the host processor. When a logical page fault occurs and the page of texture is in the second level of memory (i.e. the host's physical memory) it will be fetched automatically by the graphics memory manager, and the host is not aware anything has happened. - paragraph beginning on line 11 of page 8 of the present application.

**Claim 2:**

**Language:**

A computer system, comprising:

a graphics accelerator unit which manages page faulting of texture data, from dedicated graphics memory into a main memory used by at least one host processor, invisibly to the host processor, except when said graphics accelerator unit calls for data which has not recently been present in said main memory.

**Summary:**

In particular, the present application discloses a computer system in which a graphics accelerator unit manages page faulting of texture data invisibly to the host processor. When a logical page fault occurs and the page of texture is in the second level of memory (i.e. the host's physical memory) it will be fetched automatically by the graphics memory manager, and the host is not aware anything has happened. For this to happen a number of automatic mechanisms must be in place:

a. Determine where the page is located in host physical memory.

b. Determine which page out of the working set (in level 1 memory) to use. In a

sample embodiment, this determination uses the least recently used algorithm.

c. Make this page the most recently used page (as well as continuing to keep the least-recently-used list up to date as other pages are used).

d. Update the page tables for the new page and remove any reference to the page just bumped out of memory (if any).

e. Download the page.

f. Restart texture processing.

Note that if the faulting logical page identifies a page in the third level memory the host does (a) (after having made the page available), but the hardware carries on and does b, c, d, e and f. - paragraph beginning on line 11 of page 8 of the present application.

**Claim 3**

**Language:**

A computer system, comprising:

at least one CPU, operatively connected to have read/write access to a main memory;

first memory management logic, which virtualizes said main memory with reference to at least one bulk storage unit; and

a graphics accelerator unit, comprising rendering accelerator logic, dedicated graphics memory, and a second memory management unit which manages texture data for said accelerator logic and performs page faulting of said texture data, invisibly to said CPU.

**Summary:**

In particular, the present application discloses a computer system in which a graphics accelerator unit manages page faulting of texture data invisibly to the host processor. When a logical page fault occurs and the page of texture is in the second level of memory (i.e. the host's physical memory) it will be fetched automatically by the graphics memory manager, and the host is not aware anything has happened. For this to happen a number of automatic mechanisms must be in place:

a. Determine where the page is located in host physical memory.

b. Determine which page out of the working set (in level 1 memory) to use. In a

sample embodiment, this determination uses the least recently used algorithm.

c. Make this page the most recently used page (as well as continuing to keep the least-recently-used list up to date as other pages are used).

d. Update the page tables for the new page and remove any reference to the page just bumped out of memory (if any).

e. Download the page.

f. Restart texture processing.

Note that if the faulting logical page identifies a page in the third level memory the host does (a) (after having made the page available), but the hardware carries on and does b, c, d, e and f.

It should be noted that, once an interrupt is issued to get memory services, what happens in hardware is not a concern for the host nor for the rendering software.

Notable (and separately innovative) features of the virtual texture mapping architecture described in the present application include at least the following: A single chip solution is provided; Two or three levels of texture memory hierarchy are supported; The page faulting is all done in hardware with no host intervention; The texture memory management function can be used to manage texture storage in the host memory in addition to the texture storage in our normal texture memory; multiple memory pools are supported; and multiple rasterizers can be supported. - paragraphs from line 11 of page 8 to line 4 of page 9 of the present application.

**Claim 4**

**Language:**

A computer system comprising:

a host processor having respective physical memory associated therewith; and

a graphics accelerator unit having respective local memory associated therewith, and also having a graphics memory manager;

wherein, when said graphics accelerator unit attempts to access texture data which is in said physical memory associated with said host, said graphics memory manager fetches said texture data automatically.

**Summary:**

In particular, the present application discloses a computer system in which a graphics accelerator unit manages page faulting of texture data invisibly to the host processor. When a logical page fault occurs and the page of texture is in the second level of memory (i.e. the host's physical memory) it will be fetched automatically by the graphics memory manager, and the host is not aware anything has happened. For this to happen a number of automatic mechanisms must be in place:

a. Determine where the page is located in host physical memory.

b. Determine which page out of the working set (in level 1 memory) to use. In a sample embodiment, this determination uses the least recently used algorithm.

c. Make this page the most recently used page (as well as continuing to keep the least-recently-used list up to date as other pages are used).

d. Update the page tables for the new page and remove any reference to the page just bumped out of memory (if any).

e. Download the page.

f. Restart texture processing.

Note that if the faulting logical page identifies a page in the third level memory the host does (a) (after having made the page available), but the hardware carries on and does b, c, d, e and f.

It should be noted that, once an interrupt is issued to get memory services, what happens in hardware is not a concern for the host nor for the rendering software.

Notable (and separately innovative) features of the virtual texture mapping architecture described in the present application include at least the following: A single chip solution is provided; Two or three levels of texture memory hierarchy are supported; The page faulting is all done in hardware with no host intervention; The texture memory management function can be used to manage texture storage in the host memory in addition to the texture storage in our normal texture memory; multiple memory pools are supported; and multiple rasterizers can be supported. - paragraphs from line 11 of page 8 to line 4 of page 9 of the present application.

**Claim 7**

**Language:**

A computer system comprising:

a host processor having host physical memory associated therewith, and also having virtual memory management; and

a graphics accelerator unit having respective physical memory associated therewith, and also having virtual memory management; and

wherein, when said graphics accelerator unit attempts to access texture data which is in said host physical memory,

if said texture data is in said host physical memory, said graphics memory manager fetches said texture data therefrom automatically;

and if said texture data is not in said host physical memory, said texture data is first loaded into said host physical memory, and thereafter said graphics memory manager fetches said texture data automatically from said host physical memory.

**Summary:**

In particular, the present application discloses a computer system in which a graphics accelerator unit manages page faulting of texture data invisibly to the host processor. When a logical page fault occurs and the page of texture is in the second level of memory (i.e. the host's physical memory) it will be fetched automatically by the graphics memory manager, and the host is not aware anything has happened. For this to happen a number of automatic mechanisms must be in place:

a. Determine where the page is located in host physical memory.

b. Determine which page out of the working set (in level 1 memory) to use. In a sample embodiment, this determination uses the least recently used algorithm.

c. Make this page the most recently used page (as well as continuing to keep the least-recently-used list up to date as other pages are used).

d. Update the page tables for the new page and remove any reference to the page just bumped out of memory (if any).

e. Download the page.

f. Restart texture processing.

Note that if the faulting logical page identifies a page in the third level memory the host

does (a) (after having made the page available), but the hardware carries on and does b, c, d, e and f.

It should be noted that, once an interrupt is issued to get memory services, what happens in hardware is not a concern for the host nor for the rendering software.

Notable (and separately innovative) features of the virtual texture mapping architecture described in the present application include at least the following: A single chip solution is provided; Two or three levels of texture memory hierarchy are supported; The page faulting is all done in hardware with no host intervention; The texture memory management function can be used to manage texture storage in the host memory in addition to the texture storage in our normal texture memory; multiple memory pools are supported; and multiple rasterizers can be supported. - paragraphs from line 11 of page 8 to line 4 of page 9 of the present application.

GROUNDS FOR REJECTION TO BE REVIEWED ON APPEAL

I. Claims 1, 3, and 13 are rejected under 35 U.S.C. §102(e) as being anticipated by Mizuyabu *et al.* (U.S. Patent No. 6,297,832).

II. Claim 12 is rejected under 35 U.S.C. §103(a) as being unpatentable over Mizuyabu *et al.* (U.S. Patent No. 6,297,832) in view of Porterfield (U.S. Patent No. 6,249,853).

III. Claims 4, 5, 7, 16, 18, and 19 are rejected under 35 U.S.C. §103(a) as being unpatentable over Peddada *et al.* (U.S. Patent No. 6,295,068) in view of Emberling *et al.* (U.S. Patent No. 6,246,422).

ARGUMENT

**I. Claims 1, 3, and 13 are rejected under 35 U.S.C. §102(e) as being anticipated by Mizuyabu *et al.* (U.S. Patent No. 6,297,832).**

**Claim 1**

**A computer system, comprising: a graphics accelerator unit which manages page faulting of texture data invisibly to the host processor.**

- ***The video graphics circuit of Mizuyabu et al. is not designed or intended to be a subcomponent of a computer system having a separate host processor.***

Mizuyabu *et al.* do not teach each element of claim 1. Specifically, claim 1 recites, "a graphics accelerator unit which manages page faulting of texture data invisibly to the host processor."

Examiner Tung has correctly noted on page 3 of the November 30, 2004, office action that the video graphics circuit of Mizuyabu *et al.* does not involve a host processor. However, Examiner Tung incorrectly concluded this to mean that the operations of the circuit are invisible to the host processor. The reason that a host processor is not shown in Fig. 1 of Mizuyabu *et al.*, or even mentioned in the detailed description, is that the video graphics circuit of Mizuyabu *et al.* is designed to be part of a "set-top box" that is utilized with a TV display. It is not designed or intended to be a subcomponent of a computer system

having a separate host processor.

The video graphics circuit of Mizuyabu *et al.* statically sequences both graphics information and video images, combines them, and then sends them to a TV display. As stated in col. 4, *ll.* 1-9 of Mizuyabu *et al.*:

> *The video graphics circuit illustrated in FIG. 1 may be included in a video graphics integrated circuit that is utilized in an application such as a set-top box. A set-top box would receive a data stream that includes both video and graphics information and process the data stream to provide a display that includes both video and graphics images. Preferably, this includes providing the video and graphics information to displays that would include a high definition television set (HDTV).*

Accordingly, the video graphics circuit of Mizuyabu *et al.* cannot be said to manage page faulting invisibly to the host processor when there is no host processor.

- ***Mizuyabu et al. have a different definition of page fault than the present application.***

Mizuyabu *et al.* use the term "page fault penalty" to describe the preparation time needed to prepare the memory to access a new page in a single bank of memory. Col. 3, *ll.* 10-16 of Mizuyabu *et al.* states:

> *In a typical memory structure, a number of pages will exist within the memory. Accesses to memory locations within a single page incur no timing penalty, however when accesses to different pages in a single bank of the memory occur, a page fault penalty is incurred. This page fault penalty is typically associated with the preparation time needed to prepare the memory to access the new page.*

By contrast, the present application describes a "page fault penalty" as having the host retrieve a page of texture from the second level of memory, *i.e.* the host's physical memory. This is described in the paragraph beginning on line 11 of page 8:

> **In particular, the present application discloses a computer system in which a graphics accelerator unit manages page faulting of texture data invisibly to the host processor. When a logical page fault**

**occurs and the page of texture is in the second level of memory (i.e. the host's physical memory) it will be fetched automatically by the graphics memory manager, and the host is not aware anything has happened.**

Accordingly, Mizuyabu *et al.* and the present inventions have differing concepts of hiding page faults. Mizuyabu *et al.* hide their page faults by simply dividing the memory in separate banks such that as one bank is incurring a page fault penalty associated with switching pages, another bank can be used to transfer data. Col. 3, *ll.* 17-19 of Mizuyabu *et al.* states, "*A page fault penalty associated with switching pages within one bank of the memory can be hidden if the other bank is simultaneously used to perform data transfer.*" Accordingly, the delay in processing a new page from the memory bank is reduced.

It is noted that Mizuyabu *et al.* do not appear to relieve any processing burden on any processor involved in the accessing of the data when a page fault occurs. In fact, by requiring a divided memory and switching between memory banks, Mizuyabu *et al.* increases the burden on a processor in order to reduce delays.

By contrast, the present inventions do not hide page faults merely by reducing the delay in processing, but by relieving the burden on the host processor. The present innovations remove the host processor from the "page fault" routine that would normally invoke the host processor. The present inventions accomplish this by retrieving data from the host's physical memory without any intervention from the host, and the host is not aware that anything has happened. This is described in the paragraphs beginning on line 11 of page 8:

**In particular, the present application discloses a computer system in which a graphics accelerator unit manages page faulting of texture data invisibly to the host processor. When a logical page fault occurs and the page of texture is in the second level of memory (i.e. the host's physical memory) it will be fetched automatically by the graphics memory manager, and the host is not aware anything has happened. For this to happen a number of automatic mechanisms must be in place:**

**a. Determine where the page is located in host physical memory.**

**b. Determine which page out of the working set (in level 1 memory) to use. In a sample embodiment, this determination uses the least recently used algorithm.**

**c. Make this page the most recently used page (as well as continuing to keep the least-recently-used list up to date as other pages are used).**
**d. Update the page tables for the new page and remove any reference to the page just bumped out of memory (if any).**
**e. Download the page.**
**f. Restart texture processing.**
**Note that if the faulting logical page identifies a page in the third level memory the host does (a) (after having made the page available), but the hardware carries on and does b, c, d, e and f.**

**It should be noted that, once an interrupt is issued to get memory services, what happens in hardware is not a concern for the host nor for the rendering software.**

Accordingly, although both Mizuyabu *et al.* and the present application describe their respective innovations in terms of page faults and hiding page faults, they do not refer to the same innovations.

A prior art reference anticipates the claimed invention under 35 U.S.C. §102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). In the present case, Examiner Tung has noted that the video graphics circuit of Mizuyabu *et al.* does not involve a host processor, and has not cited teaching in the reference that discloses each and every element of claim 1. Accordingly, Applicant respectfully submits that claim 1 is not anticipated by Mizuyabu *et al.*

**Claim 3**

**A computer system, comprising:**

**at least one CPU, operatively connected to have read/write access to a main memory;**

**first memory management logic, which virtualizes said main memory with reference to at least one bulk storage unit; and**

**a graphics accelerator unit, comprising rendering accelerator logic, dedicated graphics memory, and a second memory management unit which manages texture**

**data for said accelerator logic and performs page faulting of said texture data, invisibly to said CPU.**

- ***Mizuyabu et al. do not disclose a CPU or a main memory.***

Mizuyabu *et al.* do not teach each element of claim 3. Specifically, claim 3 recites, "at least one CPU, operatively connected to have read/write access to a main memory."

Examiner Tung has correctly noted on page 3 of the November 30, 2004, office action that Mizuyabu *et al.* do not show a CPU. Examiner Tung goes on to suggest that, "it is inherent to any well known computer system to include at least one CPU." However, as established earlier, the video graphics circuit of Mizuyabu *et al.* is designed to be part of a "set-top box" that is utilized with a TV display. It is not intended or designed to be a subcomponent of a computer system. Therefore, a CPU would not be inherent in the video graphics circuit of Mizuyabu *et al.*

Examiner Tung also has correctly noted on page 4 of the November 30, 2004, office action that Mizuyabu *et al.* do not show a main memory. Again, Examiner Tung suggests that a main memory is inherent to any well known computer system. However, again, the video graphics circuit of Mizuyabu *et al.* is designed to be part of a "set-top box" that is utilized with a TV display. It is not intended or designed to be a subcomponent of a computer system. Therefore, a main memory would not be inherent in the video graphics circuit of Mizuyabu *et al.*

Accordingly, the most obvious reason that neither a CPU nor even a main memory is shown or discussed in Mizuyabu *et al.* is that video graphics circuit of Mizuyabu *et al.* is not designed or intended to be part of a computer system having a CPU and main memory.

- ***Mizuyabu et al. do not disclose a first memory management logic or a bulk storage unit.***

Claim 3 also recites, "first memory management logic, which virtualizes said main memory with reference to at least one bulk storage unit."

Examiner Tung has correctly noted on page 4 of the November 30, 2004, office action that Mizuyabu *et al.* do not show a first memory management logic. Again, Examiner Tung suggests that it is, "inherent to any well known computer system to include MMU to

manage virtual memory." However, as established above, the video graphics circuit of Mizuyabu *et al.* is not designed to be a subcomponent of a computer system. Therefore, memory management logic would not be inherent in the video graphics circuit of Mizuyabu *et al.*

Also, the video graphics circuit of Mizuyabu *et al.* does not have virtual memory because it relates to the static sequencing of memory accesses in a video graphics system by dividing the memory into two banks. As stated in col. 2, *ll.* 15-34 of Mizuyabu *et al.*:

> *Generally, the present invention provides a method and apparatus for sequencing memory accesses in a video graphics system such that page faults are effectively hidden. This is accomplished by receiving a memory access request from one of a plurality of clients, where the plurality of clients includes both linear clients and tiled memory clients. The clients access data stored in a memory that includes at least two banks. Once the memory request has been received, it is evaluated based on other pending requests in order to determine the optimal ordering pattern for execution of the memory requests. The optimal ordering pattern typically includes sequencing alternating accesses between the two banks of the memory such that when a page fault is occurring in one bank of the memory, a memory access is occurring in the opposing bank. By ensuring that an effective access takes place in one bank while the other bank prepares to access memory that is incurring a page fault, efficiency of the memory accesses can be greatly enhanced. Once the ordering of the memory requests has been performed, the requests are executed.*

Virtual memory, for example, uses disk storage space to make the computer work as if it had more memory. When a file or program is too big for the computer to work with in its memory, part of the data is stored on disk. This virtual storage is divided into segments called pages; each page is correlated with a location in physical memory, or RAM. When an address is referenced, the page is swapped into memory; it is sent back to disk when other pages must be called. This allows the program to run as if all the data is in memory.[1]

The static sequencing of memory accesses between two memory banks does not require virtual memory as evidenced by the fact that virtual memory is not even mentioned in the Mizuyabu *et al.* patent. Examiner Tung has noted that the reason for

---

[1] This particular teaching is illustrated at http://www.computeruser.com/resources/dictionary/definition.html?lookup=5891.

having an MMU is to manage virtual memory. Accordingly, there would be no need for memory management logic if there is no virtual memory to manage. Therefore, Examiner Tung's suggestion that an MMU would be inherent in the video graphics circuit of Mizuyabu *et al.* is incorrect.

Also, Mizuyabu *et al.* do not teach or suggest a bulk storage unit. Examiner Tung has failed to show how this element is taught by Mizuyabu *et al.* With regard to this element, Examiner Tung merely suggests on page 4 of the November 30, 2004, "such, as, disk memory, by definition from Microsoft Press Computer Dictionary, virtual memory may be partially simulated by secondary storage such as a hard disk." Mizuyabu *et al.* do not teach or suggest disk memory or virtual memory.

- ***Mizuyabu et al. do not teach performing page faulting invisibly to a CPU.***

Claim 3 also recites, "a graphics accelerator unit, comprising rendering accelerator logic, dedicated graphics memory, and a second memory management unit which manages texture data for said accelerator logic and performs page faulting of said texture data, invisibly to said CPU."

As established earlier, the video graphics circuit of Mizuyabu *et al.* is designed to be part of a "set-top box" that is utilized with a TV display. Therefore, a CPU would not be inherent in the video graphics circuit of Mizuyabu *et al.* Accordingly, Mizuyabu *et al.* would not teach performing page faulting invisibly to a CPU that is not disclosed by or inherent in the video graphics circuit of Mizuyabu *et al.*

Therefore, for the reasons stated above, Examiner Tung has not cited teaching in the reference that discloses each and every element of claim 3. Accordingly, Applicant respectfully submits that claim 3 is not anticipated by Mizuyabu *et al.*

**Claim 13**

**The system of Claim 3, wherein said bulk storage unit is a mass storage disk drive.**

Dependent claim 13, which depends directly from independent claim 3 and

incorporates all the limitations thereof, also includes additional limitations that are not shown or suggested by Mizuyabu *et al.* Specifically, claim 13 recites, "wherein said bulk storage unit is a mass storage disk drive." As established earlier, Mizuyabu *et al.* do not teach or suggest a bulk storage unit, much less a mass storage disk drive.

Therefore, for the reasons stated above, Applicant respectfully submits that claim 13 is also not anticipated by Mizuyabu *et al.*

Thus, for the reasons discussed above, Applicant respectfully requests reversal of this rejection.

**II. Claim 12 is rejected under 35 U.S.C. §103(a) as being unpatentable over Mizuyabu *et al.* (U.S. Patent No. 6,297,832) in view of Porterfield (U.S. Patent No. 6,249,853).**

**Claim 12**

**The system of Claim 3, wherein said first memory management logic is a bridge controller.**

- ***Mizuyabu et al. do not teach a first memory management logic, much less a first memory management logic that is a bridge controller.***

Dependent claim 12, which depends directly from independent claim 3 and incorporates all the limitations thereof, also includes additional limitations that are not shown or suggested by Mizuyabu *et al.* Specifically, claim 12 recites, "wherein said first memory management logic is a bridge controller." As established earlier, Mizuyabu *et al.* do not teach or suggest a first memory management logic, much less a first memory management logic that is a bridge controller. Accordingly, even if one were motivated to combine the teachings of Porterfield with the video graphics circuit of Mizuyabu *et al.* (which Applicant strongly disputes), it still would not result in the present inventions.

All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir.

1994). Therefore, a prima facie case of obviousness has not been established by Examiner Tung with regard to claim 12. Accordingly, Applicant respectfully submits that Claim 12 is not obvious in view of Mizuyabu *et al.* and Porterfield, and respectfully requests reversal of this rejection.

**III. Claims 4, 5, 7, 16, 18, and 19 are rejected under 35 U.S.C. §103(a) as being unpatentable over Peddada *et al.* (U.S. Patent No. 6,295,068) in view of Emberling *et al.* (U.S. Patent No. 6,246,422).**

**Claim 4**

**A computer system comprising:**

**a host processor having respective physical memory associated therewith; and**

**a graphics accelerator unit having respective local memory associated therewith, and also having a graphics memory manager;**

**wherein, when said graphics accelerator unit attempts to access texture data which is in said physical memory associated with said host, said graphics memory manager fetches said texture data automatically.**

- ***Peddada does not teach a graphics memory manager.***

The asserted combination of references does not teach or suggest each and every element of claim 4. Specifically, claim 4 recites, "a graphics accelerator unit having respective local memory associated therewith, and also having a graphics memory manager; wherein, when said graphics accelerator unit attempts to access texture data which is in said physical memory associated with said host, said graphics memory manager fetches said texture data automatically."

Examiner Tung has correctly noted on page 5 of the November 30, 2004, office action that, "Peddada fails to explicitly teach the graphics accelerator unit also having a graphics memory manager." Therefore, there is no graphics memory manager in the graphics accelerator unit of Peddada to automatically fetch texture data from the physical

memory associated with the host.

- ***Emberling et al. do not teach or suggest a memory manager.***

Examiner Tung appears to suggest that storage logic 64 of Emberling *et al.* is a memory manager. However, Applicant respectfully disagrees with this suggestion. Storage logic 64 appears to do nothing more than to receive large mip maps, split them into smaller portions, and store various portions of mip maps into predefined memory banks. This understanding is supported by the claim language used to describe the function of storage logic 64. For example, claim 14 of Emberling *et al.* states:

> *wherein said storage logic is configured to receive a series of mip maps from said main system memory, wherein successive mip maps of said series have successively smaller sizes, wherein the series of mip maps include large mip maps having sizes larger than a page and small mip maps having sizes less than or equal to a page, wherein the storage logic is configured to (a) split each of said large mip maps into a first portion and a second portion, (b) store the first portions of any two consecutive large mip maps into distinct memory banks of said texture memory, (c) store the second portions of any two consecutive large mip maps into distinct memory banks of said texture memory, (d) store small mip maps, after a first small mip map in said series, into a first memory bank of said texture memory.*

Similar descriptions of storage logic 64 can be found in claims 19, 22, and 27 of Emberling *et al.*

"Memory manager" is a term of art used, for example, to describe a part of a computer program that accepts requests from the program to allocate and deallocate chunks of memory. The objectives of the memory manager are generally to allow dynamic memory allocation.[2] Storage logic 64 does not perform such functions. Therefore, it could not properly be considered to be a memory manager.

Even if one were to assume that storage logic 64 is a memory manager (which Applicant strongly disputes), storage logic 64 does not automatically fetch texture data from the host's physical memory. It merely receives the mip maps from the host processor and splits them into smaller mip maps. Col. 8, *ll.* 26-33 of Emberling *et al.* states:

> *Each polygon conveyed to graphics subsystem 112 is conveyed to*

---

[2] This particular teaching is illustrated at http://encyclopedia.thefreedictionary.com/Memory%20manager.

*a graphics processing core 66 for texture mapping and rendering. The texture map is conveyed to storage logic 64 for eventual storage in texture memory 40 as described above. In one embodiment, host CPU 102 calculates a series of mip maps from the texture map and conveys these to storage logic 64 in order from large to small.*

Hence, even if one were to combine the teachings of Peddada *et al.* with the teachings of *Emberling et al.*, it still would not result in the present inventions as Emberling *et al.* does not disclose a graphics memory manager that automatically fetches texture data from the main memory. Therefore, a prima facie case of obviousness has not been established by Examiner Tung with regard to claim 4. Accordingly, for the reasons stated above, Applicant respectfully submits that claim 4 is not obvious in view of Peddada *et al.* and Emberling *et al.*

**Claim 7**

**A computer system comprising:**

**a host processor having host physical memory associated therewith, and also having virtual memory management; and**

**a graphics accelerator unit having respective physical memory associated therewith, and also having virtual memory management; and**

**wherein, when said graphics accelerator unit attempts to access texture data which is in said host physical memory,**

**if said texture data is in said host physical memory, said graphics memory manager fetches said texture data therefrom automatically;**

**and if said texture data is not in said host physical memory, said texture data is first loaded into said host physical memory, and thereafter said graphics memory manager fetches said texture data automatically from said host physical memory.**

The asserted combination of references also does not teach or suggest each and every element of claim 7. Specifically, claim 7 recites, "a graphics accelerator unit having respective physical memory associated therewith, and also having virtual memory management; and wherein, when said graphics accelerator unit attempts to access texture

data which is in said host physical memory, if said texture data is in said host physical memory, said graphics memory manager fetches said texture data therefrom automatically; and if said texture data is not in said host physical memory, said texture data is first loaded into said host physical memory, and thereafter said graphics memory manager fetches said texture data automatically from said host physical memory."

As stated earlier, Examiner Tung has correctly noted that, "Peddada fails to explicitly teach the graphics accelerator unit also having a graphics memory manager." Therefore, there is no graphics memory manager in the graphics accelerator unit of Peddada to automatically fetch texture data from the physical memory associated with the host.

Again, Examiner Tung appears to suggest that storage logic 64 of Emberling *et al.* is a memory manager. For the reasons stated above, Applicant respectfully disagrees with this suggestion.

Neither of the applied references, or any motivated combination thereof, discloses or suggests a graphics accelerator unit having virtual memory management with a graphics memory manager capable of automatically fetching texture data from the host's physical memory. Therefore, a prima facie case of obviousness has not been established by Examiner Tung with regard to claim 7. Accordingly, Applicant respectfully submits that claim 7 is not obvious in view of Peddada *et al.* and Emberling *et al.*

**Claims 5, 16, 18, and 19**

Dependent claims 5, 16, 18, and 19, which depend directly from independent claims 4 and 7 and incorporate all the limitations thereof, also include additional limitations that are not shown or suggested by applied references.

Specifically, claim 5 recites, "wherein, after fetching said texture data, said graphics memory manager restarts texture processing." Neither of the applied references, or any motivated combination thereof, teaches or suggests a graphics memory manager that automatically fetches texture data, much less one that restarts the texture processing after it fetches the texture data.

Claim 18 recites, "wherein said graphics memory manager comprises: means for determining where a page is located in host physical memory; means for updating page tables for said page; means for downloading said page; and means for restarting texture

processing." As established earlier, neither of the applied references, or any motivated combination thereof, teaches or suggests a graphics memory manager that automatically fetches texture data, much less one that determines where a page is located in host physical memory, updates page tables, downloads the page, and restarts the texture processing.

Therefore, for the reasons stated above, Applicant respectfully submits that claims 5, 16, 18, and 19 also are not obvious in view of Peddada *et al.* and Emberling *et al.*

Thus, for the reasons discussed above, Applicant respectfully requests reversal of this rejection.

Conclusion

For the reasons advanced above, Appellant respectfully contends that each claim is patentable. Therefore, reversal of all rejections is respectfully requested.

Respectfully submitted,

04/22/2005

Date

N. Elizabeth Pham
Attorney for Applicant
Registration No. 49,042

**GROOVER & HOLMES**
**Customer No. 29106**
P.O. Box 802889
Dallas TX 75380-2889
Tel: 972-980-5840
Fax: 972-980-5841

## CLAIMS APPENDIX

1. A computer system, comprising:

a graphics accelerator unit which manages page faulting of texture data invisibly to the host processor.

2. A computer system, comprising:

a graphics accelerator unit which manages page faulting of texture data, from dedicated graphics memory into a main memory used by at least one host processor, invisibly to the host processor, except when said graphics accelerator unit calls for data which has not recently been present in said main memory.

3. A computer system, comprising:

at least one CPU, operatively connected to have read/write access to a main memory;

first memory management logic, which virtualizes said main memory with reference to at least one bulk storage unit; and

a graphics accelerator unit, comprising rendering accelerator logic, dedicated graphics memory, and a second memory management unit which manages texture data for said accelerator logic and performs page faulting of said texture data, invisibly to said CPU.

4. A computer system comprising:

a host processor having respective physical memory associated therewith; and

a graphics accelerator unit having respective local memory associated therewith, and also having a graphics memory manager;

wherein, when said graphics accelerator unit attempts to access texture data which is in said physical memory associated with said host, said graphics memory manager fetches said texture data automatically.

5. The system of Claim 4, wherein, after fetching said texture data, said graphics memory manager restarts texture processing.

6. (canceled).

7. A computer system comprising:

a host processor having host physical memory associated therewith, and also having virtual memory management; and

a graphics accelerator unit having respective physical memory associated therewith, and also having virtual memory management; and

wherein, when said graphics accelerator unit attempts to access texture data which is in said host physical memory,

if said texture data is in said host physical memory, said graphics memory manager fetches said texture data therefrom automatically;

and if said texture data is not in said host physical memory, said texture data is first loaded into said host physical memory, and thereafter said graphics memory manager fetches said texture data automatically from said host physical memory.

8. The system of Claim 2, wherein said graphics accelerator unit also includes a PCI/AGP interface, DMA controllers, SGRAM/SDRAM, a RAMDAC, and a video stream interface.

9. The system of Claim 2, wherein said dedicated graphics memory is SGRAM/SDRAM to which the unit has read-write access through its frame buffer and local buffer ports.

10. The system of Claim 2, wherein said host processor is operatively connected to receive inputs from input devices through an interface manager chip which provides an interface to various ports and registers.

11. (canceled).

12. The system of Claim 3, wherein said first memory management logic is a bridge controller.

13. The system of Claim 3, wherein said bulk storage unit is a mass storage disk drive.

14. (allowed).

15. (allowed).

16. The system of Claim 4, wherein said host processor is operatively connected to receive inputs from input devices through an interface manager chip which provides an interface to various ports and registers.

17. (allowed).

18. The system of Claim 4, wherein said graphics memory manager comprises:

means for determining where a page is located in host physical memory;

means for updating page tables for said page;

means for downloading said page; and

means for restarting texture processing.

19. The system of Claim 7, wherein said host processor is operatively connected to receive inputs from input devices through an interface manager chip which provides an interface to various ports and registers.

20. (allowed).